Open access • Journal Article • DOI:10.1109/TII.2006.875511

Providing Real-Time Applications With Graceful Degradation of QoS and Fault Tolerance According to (m, k) -Firm Model — Source link

Jian Li, Y. Song, F. Simonot-Lion

Published on: 15 May 2006 - IEEE Transactions on Industrial Informatics (IEEE)

Topics: Fault tolerance and Scheduling (computing)

Providing Real-time Applications with Graceful Degradation of QoS and Fault Tolerance According to (m,k)-firm Model

Jian Li, Ye-Qiong Song, Françoise Simonot-Lion

Providing Real-Time Applications With Graceful Degradation of QoS and Fault Tolerance According to (m, k)-Firm Model

Jian Li, YeQiong Song, and Françoise Simonot-Lion

***Abstract*—The (m, k)-firm model has recently drawn a lot of attention. It provides a flexible real-time system with graceful degradation of the quality of service (QoS), thus achieving the fault tolerance in case of system overload. In this paper, we focus on the distance-based priority (DBP) algorithm as it presents the interesting feature of dynamically assigning the priorities according to the system's current state (QoS-aware scheduling). However, DBP cannot readily be used for systems requiring a deterministic (m, k)-firm guarantee since the schedulability analysis was not done in the original proposition. In this paper, a sufficient schedulability condition is given to deterministically guarantee a set of periodic or sporadic activities (jobs) sharing a common non-preemptive server. This condition is applied to two case studies showing its practical usefulness for both bandwidth dimensioning of the communication system providing graceful degradation of QoS and the task scheduling in an in-vehicle embedded system allowing fault tolerance.**

***Index Terms*—(m, k)-firm, non-preemptive scheduling, quality of service (QoS), real-time.**

I. Introduction

It IS well known that real-time systems designed according to the worst-case condition (case of hard-real-time system design) often result in a large resource requirement. As at run time, the system is seldom in a worst-case condition, a large amount of system resources is under-utilized. One solution is to design the system based on an average case. This solution can be suitable for a subclass of soft real-time systems requiring only statistic deadline guarantee. However, for other real-time systems, such as those found in multimedia and the automatic control domain, providing only a statistic deadline guarantee can be unacceptable. A more precise specification on how the deadline misses are distributed in time is necessary [1], and this can be done using the (m, k)-firm model [2]. Typically, for the same deadline miss ratio, a real-time application better tolerates non-consecutive deadline misses than consecutive ones. A system is said under the (m, k)-firm real-time constraint if it requires the guarantee of the deadline meet of at least m out of any $k$consecutive invocations of a recurrent job.

Much previous work has dealt with new scheduling algorithms integrating the additional (m, k)-firm constraint [3]. Two families can be found: dynamic and static. Distance-based priority (DBP) [2] and dynamic window-constrained scheduling (DWCS) [4] are dynamic. The priority assignment done online is based on the current state of the system. Enhanced rate monotonic (ERM) [5] and enhanced fixed-priority (EFP) [6] are static as the scheduling is done offline using a static deadline miss pattern. These four algorithms will be briefly discussed in Section II. Finally, note that, as for hard real-time, sufficient conditions of feasibility are obviously required in order to ensure a deterministic (m, k)-firm guarantee. There are sufficient conditions for ERM, EFP, and DWCS [4], [7], but no such condition has been investigated for DBP.

In this paper, we only consider the dynamic (m, k)-firm scheduling algorithms for the following reasons. The system should be able to adapt to workload variation (e.g., in networks handling quality of service (QoS) with connection admission control) by taking advantage of the possibility to discard until $k-m$ out of k consecutive jobs during system overload periods. So, in this context, offline scheduling is simply not suitable. Furthermore, the use of a dynamic scheduling policy rather than a static one allows a better exploitation of the available resources in general. Finally, we insist on the importance of discarding the instances of jobs whose deadlines cannot be met by the system. In fact, an overload situation leads to deadline misses, and only discarding part of jobs (preferably those with missed deadlines) allows better managing of it. Scheduling with dynamic job drops makes our work different to the classic scheduling studies without drops (e.g., [1], [8], [9]).

Once we have established that a dynamic policy is better suited to the application requirements, we have to justify the choice of DBP in our work as opposed to DWCS. We recall that for the targeted applications, we have to exhibit at least a sufficient feasibility condition. For DWCS, such a condition was established in [4], but it has a limited application region since the jobs must be with the same service time and the same periods. That is why, although DBP itself could be improved [10], we investigated this scheduling in order to find a more general condition. Moreover, as we would like to obtain a result applicable to both CPU task scheduling and network packet scheduling, we further restrict ourselves to non-preemptive scheduling. As proposed in former studies [2], [4], we consider EDF for equal priority cases.

Therefore, in this paper, we focus on only non-preemptive—distance-based priority—earliest deadline first

Manuscript received November 18, 2005; revised January 27, 2006. This work was supported in part by the REMPLI EU project (http://www.rempli.org). It was presented in part at the IEEE 5th International Workshop on Factory Communication Systems, Vienna, Austria, September 22–24, 2004.

The authors are with the LORIA, Laboratoire Lorrain de Recherche en Informatique et ses Applications, Campus Scientifique, BP 239 - 54506 Vandoeuvre-lès-Nancy, France (e-mail: Jian.Li@loria.fr; Song@loria.fr; Francoise.Simonot@loria.fr).

Digital Object Identifier 10.1109/TII.2006.875511

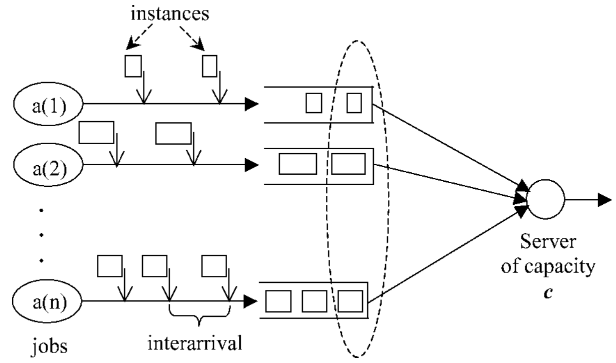

Fig. 1. MIQSS model.

(NP-DBP-EDF). The rest of this paper is organized as follows: Section II describes the problem and outlines the related work. Section III presents the sufficient schedulability condition under NP-DBP-EDF, which is the main contribution of this paper. In Section IV, we demonstrate how this condition can assist the designer for efficiently dimensioning a system. The results obtained in two case studies are compared with those obtained, within the limits of (k,k)-firm (or equivalently hard real-time), from the sufficient condition presented in [11]. The limits of the deterministic (m,k)-firm guarantee are also discussed, highlighting the need for another real-time constraint model. Finally, we conclude our findings in Section V.

II. Problem Description and Related Work

A. System Model

Consider the following multiple input queues single server (MIQSS) model (see Fig. 1), where a set of n jobs (or streams) $a() = \{a(1), a(2), \ldots, a(n)\}$ share a single server of capacity c.

We consider the jobs that are periodic or sporadic with p_i as the period or minimum inter-arrival time and a deadline $d_i = p_i$. An instance (or invocation) of $a(i)$ requires an execution time of c_i. Each job $a(i)$ is assumed to be under a (m_i, k_i)-firm constraint $(m_i < k_i)$. So, a job $a(i)$ can be modeled by (c_i, p_i, m_i, k_i).

A job under (m,k)-firm constraint can be found in one of the two following states: *normal* and *dynamic failure* [2]. To find out the current state of a job, we need to examine the execution history of the last k instances. If we associate "1" to an instance with a deadline met and "0" to an instance with a deadline miss, this history is then entirely described by a group of k bits called *k-sequence*. The system fails into a dynamic failure state when any job's k-sequence contains less than m "1." Fig. 2 shows the state transition diagram for (2,3)-firm; the left-most bit in the group represents the oldest event. Each new instance arrival causes a shift to the left in the group, the left-most exits from the k-sequence and is no longer considered, while the right-most will be a 1 if the instance has met its deadline or a 0 if not.

B. DBP Strategy

DBP together with the concept of (m,k)-firm was first introduced by Hamdaoui and Ramanathan [2] for scheduling a set of job streams sharing a common server. DBP dynamically assigns priority to the jobs of a stream according to the *distance* of the current k-sequence to a dynamic failure state. The closer the stream to a failure state, the higher its priority. This distance can be easily evaluated, by adding 0's to the right side until the failure state and the number of added 0's is equal to the priority. The k-sequence can be considered, in a way, as a kind of online QoS measurement system, and thus, DBP can be seen as a dynamic scheduling policy with feedback. The resulting priority then contributes to maintaining the global performance of the system. Results obtained from simulation [2], [10], [12] have shown that DBP provides good statistic performance, which can be used for applications requiring a statistic (m,k)-firm guarantee. However, for applications requiring a deterministic (m,k)-firm guarantee, we need a sufficient schedulability condition.

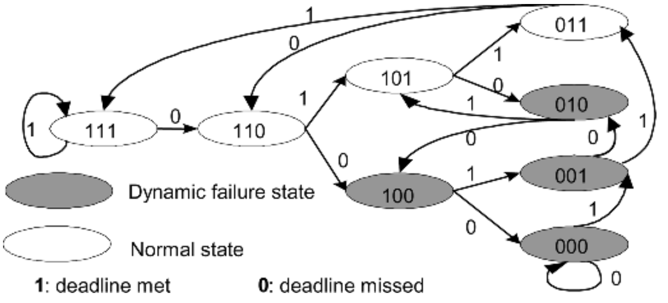

Fig. 2. State-transition diagram with (2,3)-firm.

C. Related Work

In [5], an offline fixed-priority algorithm called ERM is proposed, and the corresponding sufficient schedulability condition is given. Instances of a job are first classified as mandatory (1) and optional (0), providing a fixed k-sequence to indicate its (m,k)-firm constraint. Nevertheless, satisfying an (m,k)-firm constraint using the fixed k-sequence is more restrictive than actually needed and could potentially result in more resource requirements. Moreover, in the MIQSS model, several k_i-sequences $(i = 1, \ldots, n)$ could concentrate their mandatory instances on the time axis, resulting in a peak workload for the server. In [6], the worst-case interference point (WCIP) is defined for a job of priority i. It is the time instant at which the maximum execution interference from a higher priority job set may occur. Then, EFP is proposed to reduce WCIP. It consists in rotating the (m,k)-sequences (or (m,k)-patterns) according to a heuristic approach. It has been shown that finding the optimal superposition of k_i-sequences is NP-hard in strong sense. It is also true for any dynamic algorithm. Thus, neither DBP nor DWCS can be optimal.

Contrarily to DBP, which only uses the distance, DWCS [4] dynamically assigns priority to job $a(i)$ based on $W_i = x_i/y_i$. It ensures that in every fixed window of y_i consecutive instances, a minimum of $y_i - x_i$ instances must meet their deadline. Otherwise, a service violation occurs (dynamic failure). In DWCS, instances will be either executed before their deadlines or dropped as in DBP. Furthermore, even though DBP works in a sliding window while DWCS does in a fixed window, they have the equivalence since they can be transformed to each other.

Mok and Wang [13] have proven that in general, DWCS can fail for arbitrarily low workload. The sufficient schedulability condition, given in [4] for DWCS, has improved the utilization factor, but jobs must have the same periods and unit size execution time.

Intuitively, DBP [2] constitutes a more efficient solution, and it potentially requires less server capacity than the static algorithms. However, the schedulability analysis of a dynamic algorithm is difficult. In [11], a necessary and sufficient schedulability condition for HRT is given for a set of periodic or sporadic jobs with arbitrary release time. A job $a(i)$ is modeled by (c_i, p_i) with $d_i = p_i$. Time is assumed discrete, and clock ticks are indexed by natural numbers. Job invocations and executions only start at the clock ticks; each of the parameters c_i and p_i is expressed as multiples of clock ticks.

Jeffay's Theorem: Let $a() = \{a(1), a(2), \ldots, a(n)\}$ be a set of *sporadic or periodic* jobs sorted in non-decreasing order by periods (i.e., for any pair of jobs $a(i)$ and $a(j)$, if $i > j$, then $p_i \leq p_j$). If $a()$ is schedulable, then

1) $\sum_{\mathrm{i}=1}^{\mathrm{n}} \frac{c_i}{p_i} \leq 1$
2) $\forall i, 1 < i \leq n; \forall L, p_1 < L < p_i \; c_i + \sum_{j=1}^{i-1} \left\lfloor \frac{L-1}{p_j} \right\rfloor c_j \leq L$, and if $a()$ satisfies conditions (1) and (2), then the non-preemptive EDF scheduling algorithm will schedule any concrete set of periodic or sporadic jobs generated from $a()$.

This result could be used to give a more restrictive sufficient condition for (m, k)-firm constraint. In fact, when m is equal to k, an (m, k)-firm constraint becomes hard real-time. In this case, DBP does not work, and only EDF is used. However, the server capacity dimensioned using this condition might be oversized, as it does not drop $k - m$ out of any k consecutive instances. So, in order to deal with this problem, we apply a rationale similar to that done in Jeffay's theorem to obtain a sufficient condition for NP-DBP-EDF. This is the purpose of the next section.

III. Sufficient Condition for NP-DBP-EDF

Unlike HRT scheduling, with (m, k)-firm scheduling, there is not a condition that is both necessary and sufficient. In [10], we have given a necessary condition for NP-DBP-EDF. However, this necessary condition only tells us that meeting all (m_i, k_i)-firm constraints is impossible if the server capacity is below a certain threshold. It does not tell us what the sufficient server capacity is for meeting all (m_i, k_i)-firm constraints. Therefore, for providing deterministic guarantee, a sufficient condition is fundamental.

A. NP-DBP-EDF Scheduling Algorithm

DBP is used to decide which one of the head-of-queue job instances should be served at first in the MIQSS model. In case of the same DBP value, EDF is used. We note by $\mathrm{DBP}_j(t)$ the DBP value of job j at time point t. Under the NP-DBP-EDF scheduling policy, at time t, the instance, which is being executed in the unique server, has highest priority because of the non-preemption. Instances of a same job are stored in a FIFO queue. The instances waiting for execution at the head of the queues at time t are served in the order of their DBP priorities. The instances with $\mathrm{DBP_j(t)} = 1$ $(\mathrm{for} j = 1, 2, n)$ must be executed before their deadlines; otherwise, their $(\mathrm{m_j}, \mathrm{k_j})$-firm constraint guarantee will be violated. Instances with $\mathrm{DBP_j(t)} > 1$ will be executed if they can have the server and the operation be completed before the deadline; otherwise, they are discarded. The fact of discarding job instances makes the following schedulability analysis different from the classic ones (e.g., those found in [1] and [9]).

B. Busy Period and Workload Evaluation

We define $\mathbf{DBP} = \mathbf{X}$ **busy period** as the time interval during which the server is occupied by, and only by, the instances of jobs whose DBP value is equal to $\mathbf{X}$.

Obviously, any missed deadline of DBP $= 1$ instance will violate the (m, k)-firm constraint. This is the reason why afterward, we will only focus on the worst-case processor demand relative to the DBP $= 1$ busy period.

According to NP-DBP-EDF scheduling, except for the running instance (non-preemption), DBP $= 1$ instances have the highest priority. So, once there are DBP $= 1$ instances, they should be executed immediately or simply wait until the end of the executing instance.

C. Theorem

Theorem: Let $a()$ be a set of periodic jobs $a() = \{a(1), a(2), \ldots, a(n)\}$ $(n > 1)$, where $a(i) = (c_i, p_i, m_i, k_i), d_i = p_i$ (see Section II-A). If the job set $a()$ satisfies the following conditions C_1 and C_2 in any time interval, then NP-DBP-EDF will schedule any concrete set of periodic jobs generated from $a()$, i.e., there will not be any violation of the $(\mathrm{m_i}, \mathrm{k_i})$-firm constraints, shown in the equation at the bottom of the next page, where U is the set of instances whose DBP $= 1$ instances occur at the same time point with the arbitrary release time, while a $() - U$ is the other instances. In the worst case, the set U can include one instance of each job and $a() - U = \emptyset$ (empty set). In practice, for a concrete job set, this worst case may never be reached. L represents the arbitrary length of time.

Proof: Assume the contrary, i.e., that $a()$ satisfies conditions C_1 and C_2 from the theorem, and that there is a concrete set of periodic jobs ω_s generated from $a()$, such that a job in ω_s falls into failure state, i.e., ω_s has violated the (m, k)-firm guarantee.

We analyze the process of falling into the failure state. Let t_d be the earliest time point where ω_s falls into failure state. Obviously, only $\mathbf{DBP} = \mathbf{1}$ **busy period** leads to an (m, k)-firm violation. Starting at time t_d, we work our way backward to discover which cases occurred relative to this last DBP $= 1$ busy period, knowing that, for all the possibilities, there are only three cases we could find, as follows.

Case 1) DBP $= 1$ busy period starts from an idle time, and all executed instances have the deadlines before t_d.

Case 2) DBP $= 1$ busy period is blocked by a DBP > 1 instance, and all executed instances have deadlines before t_d.

Case 3) There are some job instances that have deadlines after t_d.

In case 1, assume that DBP $= 1$ instance appears at one time point, and it can have the server immediately. The workload

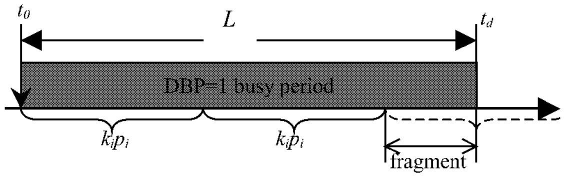

Fig. 3. Workload of DBP $=1$ instances starting at time point t_0.

is calculated in the following DBP $=1$ busy period, giving the worst-case possibility. Let t_0 be the starting time of this DBP $=1$ busy period in this situation, t_d the ending time, and let $L=t_d-t_0$ be the length of the DBP $=1$ busy period.

Jobs are divided into two sets: one is for the jobs whose DBP $=1$ instances start from the beginning time of DBP $=1$ busy period, denoted by U. Another set is $a()-U$.

As shown in Fig. 3, it is given that in every interval k_ip_i for the job $a(i)\in U$, there are m_i and only m_i instances of job $a(i)$ with DBP $=1$. Only these instances can be executed and meet their deadlines. This generates a workload of

$$W_U^1=\sum_{i\in U}\left(\left\lfloor\frac{L}{k_ip_i}\right\rfloor m_i\right)c_i \tag{1}$$

but, in general, interval L is not an integer multiple of k_ip_i. So, in the *fragment* (the residue of L divided by k_ip_i), for a job $a(i)$, the number of possible instances is bounded by m_i. This results in the following term:

$$W_U^2=\sum_{i\in U}\mathrm{Min}\left(\left\lfloor\frac{L-k_ip_i\left\lfloor\frac{L}{k_ip_i}\right\rfloor}{p_i}\right\rfloor,m_i\right)c_i. \tag{2}$$

By using (1) and (2), the workload caused by all jobs in U is then

$$W_U=W_U^1+W_U^2. \tag{3}$$

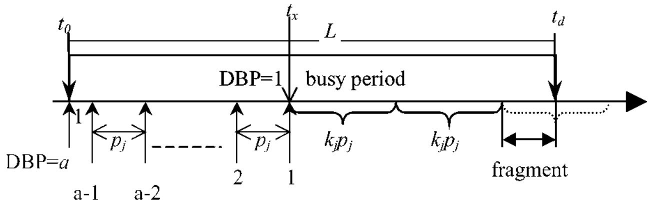

Fig. 4. Workload of instances whose DBP >1 at time point t_0.

The workload caused by the second part $(a(j)\in a()-U)$ is calculated as follows. Jobs not included in set U have their DBP value greater than 1 at the time point t_0. It is clear that in DBP $=1$ busy period, only DBP $=1$ instances can be executed. So, Fig. 4 shows how a job $a(j)$ starts to generate the workload from t_0 in DBP $=1$ busy period.

Assume that, at t_0, the job $a(j)$ has DBP $=a(a>1)$. The worst case is the following situation: after one clock tick, this DBP value will be decreased by one, and then, after every period p_j, the DBP value will be decreased by one. No instance is executed in the interval $[\mathrm{t}_0,\mathrm{t}_x]$, where t_x is the time at which $a(j)$ has its DBP $=1$. We note $l=t_x-t_0$. The worst case corresponds to

$$l=1+(\mathrm{DBP}_j(t_0)-2)p_j. \tag{4}$$

For the interval after t_x, the workload evaluation is similar to the one used for the set U. According to (1) and (2), we obtain

$$\begin{aligned}&W_{a()-U}^1\\&=\sum_{j\in a()-U}\left(\left\lfloor\frac{L-l}{k_jp_j}\right\rfloor m_j\right)c_j\end{aligned} \tag{5}$$

$$\begin{aligned}&W_{a()-U}^2\\&=\sum_{j\in a()-U}\mathrm{Min}\left(\left\lfloor\frac{(L-l)-k_jp_j\left\lfloor\frac{L-l}{k_jp_j}\right\rfloor}{p_j}\right\rfloor,m_j\right)c_j.\end{aligned} \tag{6}$$

$$\begin{aligned}\mathrm{C}_1:&\sum_{i\in U}\left(\left\lfloor\frac{L}{k_ip_i}\right\rfloor m_i+\mathrm{Min}\left(\left\lfloor\frac{L-k_ip_i\left\lfloor\frac{L}{k_ip_i}\right\rfloor}{p_i}\right\rfloor,m_i\right)\right)c_i\\&+\sum_{j\in a()-U}\left(\begin{array}{l}\left\lfloor\frac{L-1-(\mathrm{DBP}_j(t)-2)p_j}{k_jp_j}\right\rfloor m_j+\\ \mathrm{Min}\left(\left\lfloor\frac{(L-1-(\mathrm{DBP}_j(t)-2)p_j)-k_jp_j\left\lfloor\frac{L-1-(\mathrm{DBP}_j(t)-2)p_j}{k_jp_j}\right\rfloor}{p_j}\right\rfloor,m_j\right)\end{array}\right)c_j\le L\\ \mathrm{C}_2:&\ \forall i,\forall L,L>\min_i(p_i)\\&\left(\left(\left\lfloor\frac{L-c_i}{k_ip_i}\right\rfloor m_i+1\right)+\mathrm{Min}\left(\left\lfloor\frac{L-c_i-\left\lfloor\frac{L-c_i}{k_ip_i}\right\rfloor k_ip_i}{p_i}\right\rfloor-1,m_i-1\right)_+\right)c_i\\&+\sum_{j\in a()-a(i)}\left(\begin{array}{l}\left\lfloor\frac{L-1-(\mathrm{DBP}_j(t)-2)p_j}{k_jp_j}\right\rfloor m_j+\\ \mathrm{Min}\left(\left\lfloor\frac{(L-1-(\mathrm{DBP}_j(t)-2)p_j)-k_jp_j\left\lfloor\frac{L-1-(\mathrm{DBP}_j(t)-2)p_j}{k_jp_j}\right\rfloor}{p_j}\right\rfloor,m_j\right)\end{array}\right)c_j\le L\end{aligned}$$

Formula (7) expresses the total workload of the jobs in the set $(a() - U)$

$$W_{a()-U} = W^1_{a()-U} + W^2_{a()-U}. \tag{7}$$

By using (3) and (7), the total workload of a DBP $=$ 1 busy period is

$$W = W_U + W_{a()-U}. \tag{8}$$

It is easy to see that W is just the left part of the condition C_1, such that if one job's (m, k)-firm constraint is violated, W has to be bigger than L. This fact contradicts with condition C_1, and the theorem is constructed for case 1.

In cases 2 and 3, we also calculate the maximum workload taking into account the blocking instance (since non-preemption). Similarly, if there is the failure state, it will contradict with condition C_2. In this paper, because of the space limitation, we only present the proof for case 1. Cases 2 and 3 are proved with the same strategy (see [14] and [24] for the whole proof).

Above all, the theorem is constructed with the contradictions in any case.

End of Proof.

Corollary: Let $a()$ be a set of sporadic jobs $a() = \{a(1), a(2), \ldots, a(n)\}(n > 1)$, where $a(i) = (c_i, p_i, m_i, k_i), d_i = p_i$. If the job set $a()$ satisfies conditions C_1 and C_2 in any time interval, then NP-DBP-EDF will be able to schedule any concrete set of *sporadic jobs* generated from $a()$, i.e., there will be not violation of the $(\mathrm{m_i}, \mathrm{k_i})$-firm constraints.

Proof: As the worst-case behavior of a sporadic job occurs when $a(i)$ behaves like a periodic job, that is, $a(i)$ was invocated every p_i time step. Remember that a sporadic job can behave as a periodic job. Therefore, if conditions C_1 and C_2 are satisfied, the NP-DBP-EDF algorithm can schedule any concrete set generated from a periodic job set. As we have defined, the arrival curve and the workload of any sporadic job set are always inferior to the periodic concrete set. Whenever a failure state happens, the two conditions have been violated. So, the conditions are also sufficient to guarantee that NP-DBP-EDF will be able to schedule any concrete set generated from a sporadic job set.

End of proof.

D. Sufficient Verification Length

As has been shown, in our sufficient condition for NP-DBP-EDF scheduling, all DBP $_j(t)$ are a function of time. Therefore, an interval is necessary to indicate the time evaluation domain. That is to say, we need a ***sufficient length*** for terminating the verification of our sufficient condition.

First, we explain the following definitions.

1) All possible DBP values for one job $a(i)$ with the (m, k)-firm constraint: All DBP values appearing in the scheduling sequence are limited to a natural number in $[0, k_i - m_i + 1]$, but not every value will appear in a concrete situation. Because the system falls into a failure state when a DBP $= 0$ instance appears, the successful sequence under consideration (no failure state contained sequence) contains DBP values that are limited in $[1, k_i - m_i + 1]$. For a job, $a(i)$ has $k_i - m_i + 1$ DBP values in a successful scheduling sequence. In any $k_i - m_i + 2$ instances of $a(i)$, there must be at least two invocations that have the same DBP value in a successful sequence.
2) For a job set with n jobs, *at one time point*, it has $\prod_{\mathrm{i}=1}^{\mathrm{n}} (\mathrm{k_i} - \mathrm{m_i} + 1)$ successful DBP configuration possibilities.
3) For a strict periodic job set, the inter-distribution of the instances reappears after each least common multiple (LCM) of $\{p_1, \ldots, p_n\}$. Suppose that the time points $t_1, t_2, \ldots, t_x (x \in N)$ are the time points with interval LCM, i.e., $t_{i+1} = t_i + \mathrm{LCM} i \in (1, \ldots, x)$. Not considering the concrete possibilities, at all time points of $t_1, t_2, \ldots, t_x$, there are at most $\prod_{\mathrm{i}=1}^{\mathrm{n}} (\mathrm{k_i} - \mathrm{m_i} + 1)$ possible successful DBP configurations. So, in $\mathrm{x} = \prod_{\mathrm{i}=1}^{\mathrm{n}} (\mathrm{k_i} - \mathrm{m_i} + 1) + 1$, there must be at least two time points where all instances of the jobs have the same DBP values. Also, our scheduling can repeat the same successive scheduling from the two time points, because at each time point $t_1, t_2, \ldots, t_x$, the inter-distribution of the instances is always the same.

Finally, we can conclude that the sufficient length $L_{\max}$ for terminating the verification of our sufficient condition (only for a strict periodic job set) is

$$\mathrm{L_{max}} = \mathrm{r} + \left(\prod_{\mathrm{i}=1}^{\mathrm{n}} (\mathrm{k_i} - \mathrm{m_i} + 1) + 1 \right) \mathrm{LCM} \tag{9}$$

where r is the last release time.

Obviously, this is a sufficient but not necessary length, because we are considering it from an aspect of permutation. Once at a time point, the DBP values of all instances are the reappearance of DBP values, which occurred at certain LCMs before, the schedulability can already be given. Since in this case, the following sequence will be the iteration of the sequence that took place between the two time points. In practice, the test can stop earlier as soon as the repetition occurs for the first time at a multiple of LCM time point.

IV. Applications of the Sufficient Condition

In this section, we apply our sufficient condition to dimensioning the sufficient server capacity for guaranteeing the (m, k)-firm constraint in contrast to that of (k, k)-firm (i.e., HRT). In practice, the dimensioning can be done not only offline but also online. For example, a network supporting real-time QoS should be based on the sufficient condition to decide the acceptance or rejection of a new job (or stream) in its connection admission control procedure; an adaptive real-time system could go from a nominal mode corresponding to (k, k)-firm to a degraded mode, still ensuring the (m, k)-firm constraint with the presence of some resource failures.

A. Overload Management in Automotive Control Applications

In this part, we show how our sufficient condition can help the dimensioning of the processor capacity in an automotive

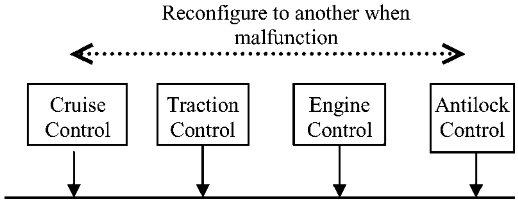

Fig. 5. Vehicle control system model.

control system for making it fault-tolerant while using reduced resources.

In in-vehicle embedded system design, the current trend is to use generic processors to replace the specific ones [15], [23]. To achieve this goal, OSEK is defined by carmakers and the electronic control unit (ECU) suppliers as the standard operating system [16]. In OSEK, the scheduling policy includes three possibilities: non-preemptive, preemptive, and the mixed one. In this case study, we assume that tasks are non-preemptive. Moreover, the effort to establish a common platform for supporting portable software modules is continuing inside the AUTOSAR consortium (see http://www.autosar.org/). One of the objectives is to be able to run a car function (e.g., engine control, ABS, etc.) over any generic processor, thus ensuring fault-tolerance when the same function is replicated on more than one processor. All the ECUs are interconnected via a bus (e.g., CAN [17], [18] or FlexRay [19] in the near future).

For making the system fault-tolerant, the classical approach consists in reserving the sufficient spare capacity so that the tasks can be reassigned or re-executed on fault-free processors upon failure detection; without violating any deadlines (i.e., (k, k)-firm). As indicated in our introduction, the drawback of this approach is that the system resources are often under-utilized when no faults are present. For the automotive industry, where the cost constraints are omnipresent, this approach has not always been acceptable. The approach based on the (m, k)-firm model is more suitable. It consists in invoking an overload management technique upon detection of a failure [5]. Following this approach, the system can still work with the presence of some processor failures without necessarily reserving as many resources as used in the classic approach.

The simulation is implemented by taking a case study similar to that of Ramanathan [5], in which the author has shown that the control laws of the automotive control applications can tolerate some deadline misses specified by the (m, k)-patterns, without leading to a dangerous situation for the vehicle. Based on our experience in automotive systems [15], [20], we add another argument that most control loops are based on over-sampling input data (sensor data) to increase dependability. The occasional loss of some input data will not automatically lead to a dangerous situation.

We then consider a system (see Fig. 5) composed of four control functions: cruise control, traction control, anti-lock braking control, and engine control. At first, all four functions are implemented on the four ECU of the system, but only one function is running on each ECU. In case of failure of an ECU, the corresponding function it ensures is woken up on one of the remaining ECU, thus tolerating an ECU failure.

TABLE I
TASK PARAMETERS OF CONTROL SYSTEM IN VEHICLE

	Execution time (ms)	Task period (ms)	(m,k)-firm constraint
Antilock control	2	20	(1,4)
Traction control	6	30	(1,4)
Engine control	5	50	(1,4)
Cruise control	6	100	(2,3)

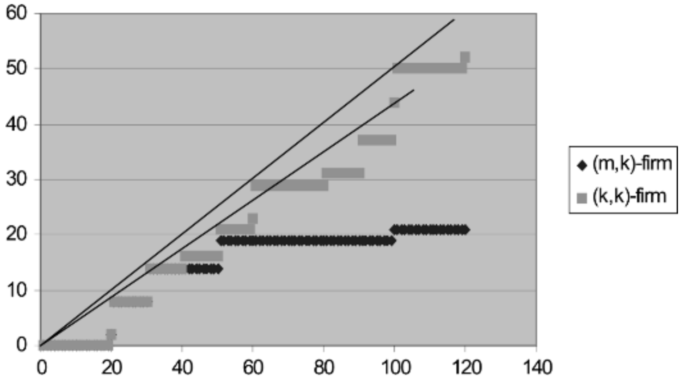

Fig. 6. Workload of (k, k)-firm in contrast to (m, k)-firm for dimensioning system sufficient capacity.

In what follows, we just consider the extreme case of three simultaneous ECU failures. Our goal is to dimension the processor capacity of an ECU to continue to guarantee meeting of the (m, k)-firm constraint of the four functions. The deadline miss tolerated by each function is assumed to be as given in Table I.

The target (m, k)-firm constraint for each function can be obtained either by following the control law stability/tolerance study method of [5] or by measuring and simulating the car situations in presence of failures (fault injection) [20].

In Fig. 6, the upper line with the slope value 0.495 is the sufficient processor capacity for the HRT measured by Jeffay's condition. The lower one with the slope value 0.42 is the sufficient processor capacity for the fault tolerant system in the form of (m, k)-firm. This represents a 15% saving of the processor capacity with respect to the original processor capacity requirement.

B. Discussion on the Limits of the Deterministic (m, k)-Firm Guarantee

As one can see from the above examples, the advantage of using (m, k)-firm, compared with (k, k)-firm, is not always noticeable. In fact, our sufficient condition and that of Jeffay can even be overlapped in some situations, thus forcing the service of all k jobs, even though the system is only under an (m, k)-firm constraint. To understand that, let us first take the following numerical example given in Table II. This configuration is derived from that of Table I with some modifications. Four streams (jobs) with $(\mathrm{m_i}, \mathrm{k_i})$-firm constraints should be executed by the MIQSS model server.

Figs. 7 and 8 give the cumulative processor demand in time for, respectively, conditions C $_1$ and C_2 in the case of HRT and (m, k)-firm of the concrete job set in Table II. The x-coordinate

TABLE II
PARAMETERS OF PERIODIC JOB SET

	(m,k)-firm constraint	Processing Time	Period/ Deadline
Stream 1	(2,5)	8	12
Stream 2	(4,5)	10	20
Stream 3	(3,6)	2	5
Stream 4	(1,5)	4	6

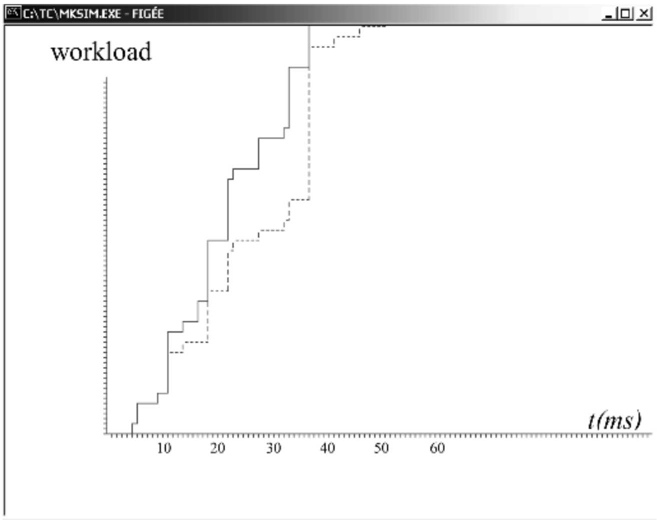

Fig. 7. Difference between conditions 1.

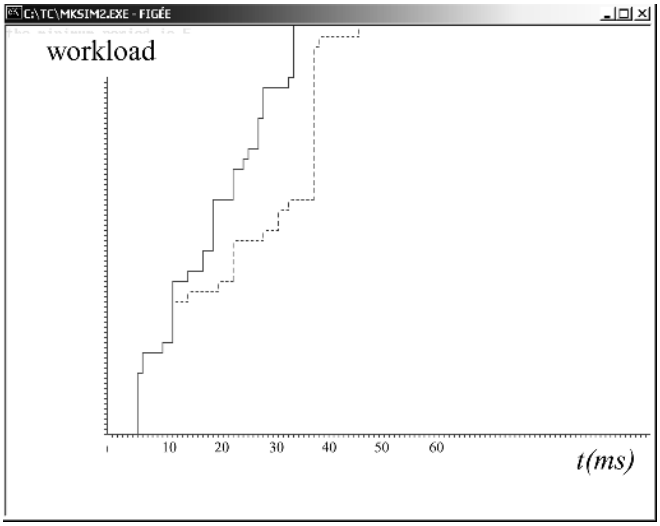

Fig. 8. Difference between conditions 2.

represents the time interval (L), and the y-coordinate represents the processor demand that must be executed before the end of L. So, we calculate the changes of this processor demand according to the length of the time interval. In Figs. 7 and 8, the upper step curve (solid line) represents the result of HRT under NP-EDF and the lower one (dashed line) that with (m,k)-firm constraint under NP-DBP-EDF. To start simulation, we assume the worst case for (m,k)-firm by setting all DBP $_i(t) = 1$.

From Figs. 7 and 8, we can see that there is an overlap at the beginning time of the simulation. In fact, condition C_1 of our theorem can be transformed to be like the condition (1) in Jeffay's theorem. Assuming that all jobs of all sources are within the set U (the worst case), and the interval L is less than $\min(\mathrm{m}_i * p_i)$, we get $\lfloor (L)/(k_ip_i)\rfloor = 0$ and the term $\mathrm{Min}(\lfloor (L - k_ip_i\lfloor (L)/k_ip_i)\rfloor)/(p_i)\rfloor, m_i)c_i = \lfloor (L)/(p_i)\rfloor c_i$. The condition C_1 of our theorem has been transformed to the condition (1) in Jeffay's theorem.

Additionally, condition C_2 of our theorem can also be transformed to be like condition (2) in Jeffay's theorem.

As we have interpreted, the sufficient condition of (m,k)-firm is always under the bound of Jeffay's theorem and can reach the bound of Jeffay's theorem with some assumptions and forced evaluation conditions. Notice that these assumptions and forced evaluation conditions can be realized or not in concrete situations. However, this limits the advantage of using the (m,k)-firm tolerance compared with a system only requiring a statistic (m,k)-firm guarantee.

We have proven in our report [21] that DBP scheduling may fail into failure state even with arbitrary low utilization. The same problem was also detected in the DWCS algorithm [13]. As in HRT, (m,k)-firm schedulability remains still NP-hard. That is why our simulation result is pessimistic. However, if we do it online within the time length of formula (9), a significant efficiency can be obtained. Furthermore, the recent proposal of [22], which relaxes the (m,k)-firm constraint by defining the virtual deadline concept, consists in an interesting way to improve the advantage of the (m,k)-firm system in terms of relaxing the resource need compared with the system requirements under (k,k)-firm.

V. CONCLUSION

In this paper, we first explained how the (m,k)-firm model can be used to define the graceful degradation of real-time QoS, thus allowing the fault-tolerance, and then addressed the problem of the deterministic guarantee of (m,k)-firm real-time requirements for a set of periodic or sporadic jobs sharing a common server. DBP has been chosen for its interesting feature of dynamically assigning priorities based on the previous history of the system (k-sequence). This makes it suitable for QoS management in adaptive real-time systems and networks. Our main contribution is having given the expression of the sufficient condition under NP-DBP-EDF scheduling for deterministically guaranteeing (m,k)-firm constraint. This result is necessary for system server capacity dimensioning. Our future work aims at two complementary directions: 1) research of conditions to avoid the overlapping of the sufficient condition of (m,k)-firm with that of (k,k)-firm and the new job models, allowing the improvement with advantages gained with (m,k)-firm in terms of relaxing resource requirements, and 2) the implementation of dynamic algorithms such as DBP, in terms of admission control procedures, within IP networks (e.g., Internet-based control systems, remote control, and monitoring systems based on Internet and power line networks, such as what has been proposed in the REMPLI project; see http://www.rempli.org) for dynamically managing real-time QoS according to the (m, k)-firm model.

REFERENCES

[1] G. Bernat, A. Burns, and A. Llamosi, "Weakly-hard real-time systems," *IEEE Trans. Comput.*, vol. 50, no. 4, pp. 308–321, Apr. 2001.

[2] M. Hamdaoui and P. Ramanathan, "A dynamic priority assignment technique for streams with (m, k)-firm deadlines," *IEEE Trans. Comput.*, vol. 44, no. 4, pp. 1443–1451, Dec. 1995.

[3] Z. Wang, Y. Q. Song, E. M. POGGI, and Y. X. Sun, "Survey of weakly-hard real-time scheduling theory and its application," in *Proc. Int. Symp. Distributed Computing Applications Business, Engineering, Science*, Wuxi, China, Dec. 16–20, 2002.

[4] R. West and C. Poellabauer, "Analysis of a window-constrained scheduler for real-time and best-effort packet streams," in *Proc. IEEE 21st Real-Time Systems Symp.*, Orlando, FL, Nov. 27–30, 2000, pp. 239–248.

[5] P. Ramanathan, "Overload management in real-time control application using (m, k)-firm guarantee," *IEEE Trans. Parallel Distrib. Syst.*, vol. 10, no. 6, pp. 549–559, Jun. 1999.

[6] G. Quan and X. Hu, "Enhanced fixed-priority scheduling with (m, k)-firm guarantee," in *Proc. IEEE 21st Real-Time Systems Symp.*, Orlando, FL, Nov. 27–30, 2000, pp. 79–88.

[7] R. West, Y. Zhang, K. Schwan, and C. Poellabauer, "Dynamic window-constrained scheduling of real-time streams in media servers," *IEEE Trans. Comput.*, vol. 53, no. 6, pp. 744–759, Jun. 2004.

[8] G. Bernat, A. Burns, and A. Llamosi, "Weakly-hard real-time systems," *IEEE Trans. Comput.*, vol. 50, no. 4, pp. 308–321, Apr. 2001.

[9] G. Bernat, "Response time analysis of asynchronous real-time systems," *Real-Time Syst.*, vol. 25, pp. 131–156, 2003.

[10] E.-M. Poggi, Y.-Q. Song, A. Koubaa, and Z. Wang, "Matrix-DBP for (m, k)-firm real-time guarantee," in *Proc. RTS*, Paris, France, Apr. 1–3, 2003, pp. 457–482.

[11] K. Jeffay, D. F. Stanat, and C. U. Martel, "On non-preemptive scheduling of periodic and sporadic task," in *Proc. IEEE Real-Time Systems Symp.*, San Antonio, TX, Dec. 4–6, 1991, pp. 129–139.

[12] F. Wang and P. Mohapatra, "Using differentiated services to support Internet telephony," *Comput. Commun.*, vol. 24, no. 18, pp. 1846–1854, Dec. 2001.

[13] A. K. Mok and W. R. Wang, "Window-constrained real-time periodic task scheduling," in *Proc. IEEE 22nd Real-Time Systems Symp.*, London, U.K., Dec. 3–6, 2001, pp. 15–24.

[14] J. Li, S. YeQiong, and S.-L. Francoise, Providing real-time applications with graceful degradation of QoS and fault tolerance according to (m, k)-firm model 2005, Rapport de recherche inria-704.

[15] C. Wilwert, N. Navet, Y. Q. Song, and F. Simonot-Lion, , R. Zurawski, Ed., "Design of automotive X-by-wire systems," in *The Industrial Communication Technology Handbook*. Boca Raton, FL: CRC, 29–1, 2005.

[16] "OSEK," in OSEK/VDX Operating System , , ver. 2.2, 2001 [Online]. Available: http://www.osek-vdx.org.

[17] "ISO," in Road Vehicles—Interchange of Digital Information—Controller Area Network for High-Speed Communication 1994, ISO 11898, international organization for standardization (ISO).

[18] L. Almeida, P. Pedreiras, and J. A. G. Fonseca, "The FTT-CAN protocol: why and how," *IEEE Trans. Ind. Electron.*, vol. 49, no. 6, pp. 1189–1201, Dec. 2002.

[19] FlexRay Consortium, 2004 [Online]. Available: http://www.flexray.com.

[20] C. Wilwert, Y. Q. Song, F. Simonot-Lion, and T. Clément, "Evaluating quality of service and behavioral reliability of steer-by-wire systems," in *Proc. IEEE 9th Int. Conf. Emerging Technologies Factory Automation*, Lisbonne, Portugal, 2003, vol. 1, pp. 193–200.

[21] J. Li, Sufficient Condition for Guaranteeing (m, k)-Firm Real-Time Requirement Under NP-DBP-EDF Scheduling 2003, Tech. Rep. No. A03-R-452, Stage de DEA, LORIA, Jun.

[22] Y. Zhang, R. West, and X. Qi, "A virtual deadline scheduler for window-constrained service guarantees," in *Proc. IEEE 25th Real-Time Systems Symp.*, Dec. 2004.

[23] P. Castelpietra, Y.-Q. Song, F. Simonot Lion, and M. Attia, "Analysis and simulation methods for performance evaluation of a multiple networked embedded architecture," *IEEE Trans. Ind. Electron.*, vol. 49, no. 6, pp. 1251–1264, Dec. 2002.

[24] L. Jian, S. YeQiong, and S.-L. Françoise, "Schedulability analysis for system under (m, k)-firm constraints," in *Proc. 5th IEEE Int. Workshop Factory Communication System*, Vienna, Austria, Sep. 2004.

Jian Li received the B.S. degree in electronic and information technology from the TianJin University, TianJin, China, in 2001 and the M.S. degree in telecommunication and computer science from the University of Henri Poincaré, Nancy, France, in 2003. He is currently pursuing the Ph.D. degree in computer science at the Institut National Polytechnique de Lorraine, Nancy.

YeQiong Song is a Professor of computer science at the Institut National Polytechnique de Lorraine, Nancy, France. His research interests include, on the one hand, the modeling and performance evaluation of networks and real-time distributed systems using queueing analysis, network calculus, and scheduling theory, and on the other hand, the implementation of real-time QoS mechanisms in fieldbuses, in-vehicle networks, switched Ethernet, IP networks, and power line communication networks.

Françoise Simonot-Lion is a Professor of computer science at the Institute National Polytechnique de Lorraine, Nancy, France. Her main research topics are modeling and verification techniques for the design of optimized real-time distributed applications under safety constraints as well as specification of embedded services ensuring a real-time quality of service (scheduling of tasks and messages, real-time middleware, and frame packing).